# An Introduction to MueLu

Trilinos User Group Meeting

November 4-8, 2013

Andrey Prokopenko

Jonathan Hu, Chris Siefert, Ray Tuminaro

2013-9485C

## Outline

- Motivation and current capabilities
- Design overview
- User interfaces and examples
- Conclusions

## Motivation and current capabilities

## Motivation for a New Multigrid Library

- Trilinos already has mature multigrid library, ML
  - Algorithms for Poisson, Elasticity, H(curl), H(div)
  - Algorithms have been exercised extensively.
  - Broad user base
- However …
  - ML weakly linked to other Trilinos capabilities (e.g., smoothers)
  - C-based, only scalar type "double" supported explicitly
  - Over 50K lines of source code
    - Maintainability, extensibility

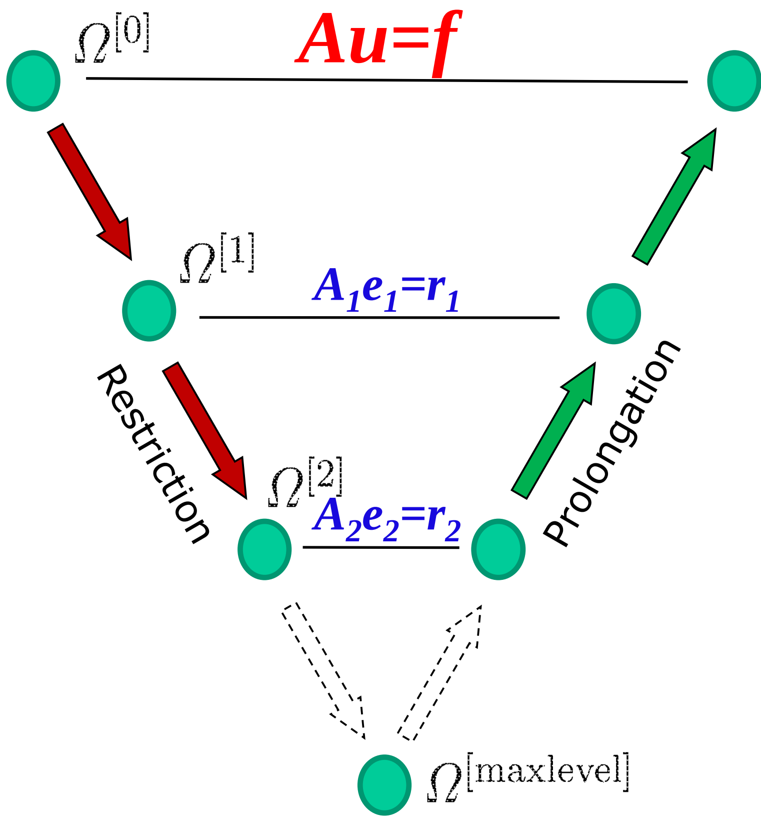
## Objectives for New Multigrid Framework

- Templating on scalar, ordinal types
- Advanced architectures
  - Kokkos support for various compute node types (MPI, MPI+threads, MPI+GPU)
- Extensibility
  - Facilitate development of other algorithms
    - Energy minimization methods
    - Geometric, classic algebraic multigrid, …
  - Ability to combine several types of multigrid
- Preconditioner reuse
  - Reduce setup expense

Sandia National Laboratories

## AMG

- **Two main components**
  - **Smoothers**
    - **Approximate solves on each level**
    - **“Cheaply” reduces particular error components**
    - **On coarsest level, smoother = $A_i^{-1}$ (usually)**
  - **Grid Transfers**
    - **Moves data between levels**
    - **Must represent components that smoothers can’t reduce**
- **Algebraic Multigrid (AMG)**
  - **AMG generates grid transfers**
  - **AMG generates coarse grid $A_i$’s**



## Current MueLu Capabilities

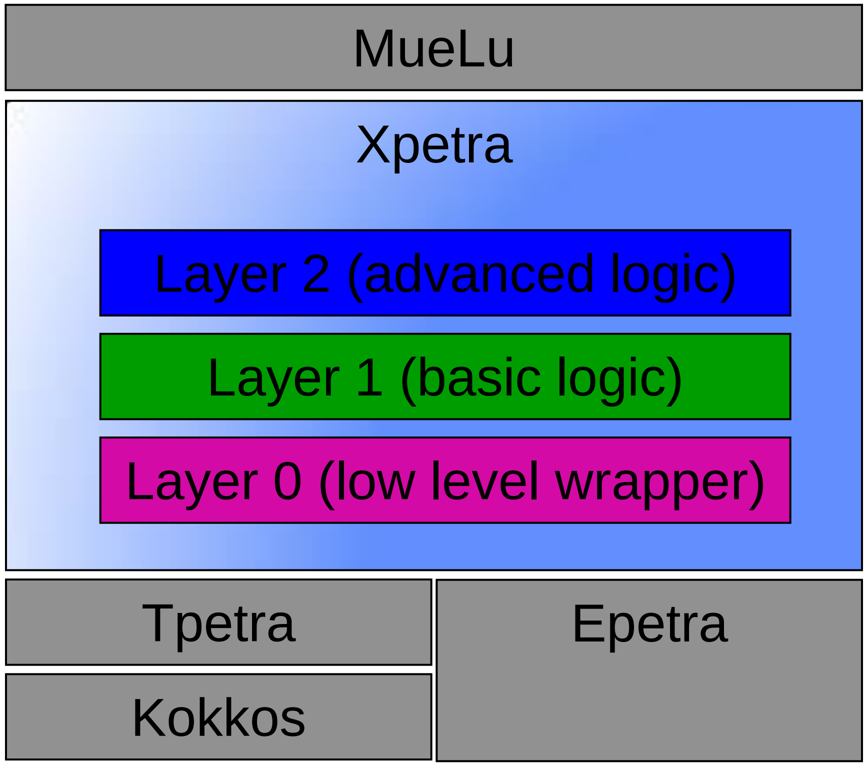
- Transfer operators
  - Smoothed aggregation
  - Nonsmoothed aggregation
  - Petrov Galerkin
  - Energy minimization
- Smoothers and direct solvers
  - Ifpack/Ifpack2 (Jacobi, Gauss-Seidel, ILU, polynomial, …)
  - Amesos/Amesos2 (KLU, Umfpack, Superlu, …)
  - Block smoothers (Braess Sarazin, …)

**We support both Epetra and Tpetra!**

Sandia National Laboratories

# Xpetra

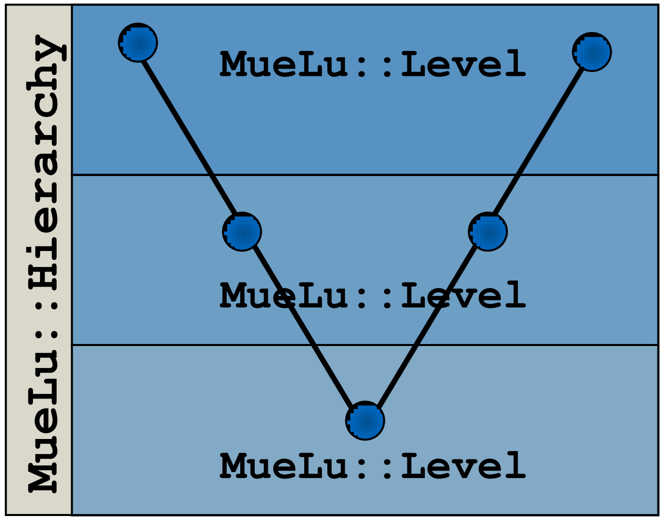
- Wrapper for Epetra and Tpetra
  - Based on Tpetra interfaces
  - Allows unified access to either linear algebra library
- Layer concept:
  - Layer 2: blocked operators
  - Layer 1: operator views
  - Layer 0: low level E/Tpetra wrappers (automatically generated code)
- MueLu algorithms are written using Xpetra



## Design overview

## Design

- **Hierarchy**
  - Generates and stores data
  - Provides multigrid cycles
- **Factory**
  - Generates data
- **FactoryManager**
  - Manages dependencies among factories



**Preconditioner is created by linking together factories (constructing FactoryManager) and generating Hierarchy data using that manager.**

**User is not required to specify these dependencies.**

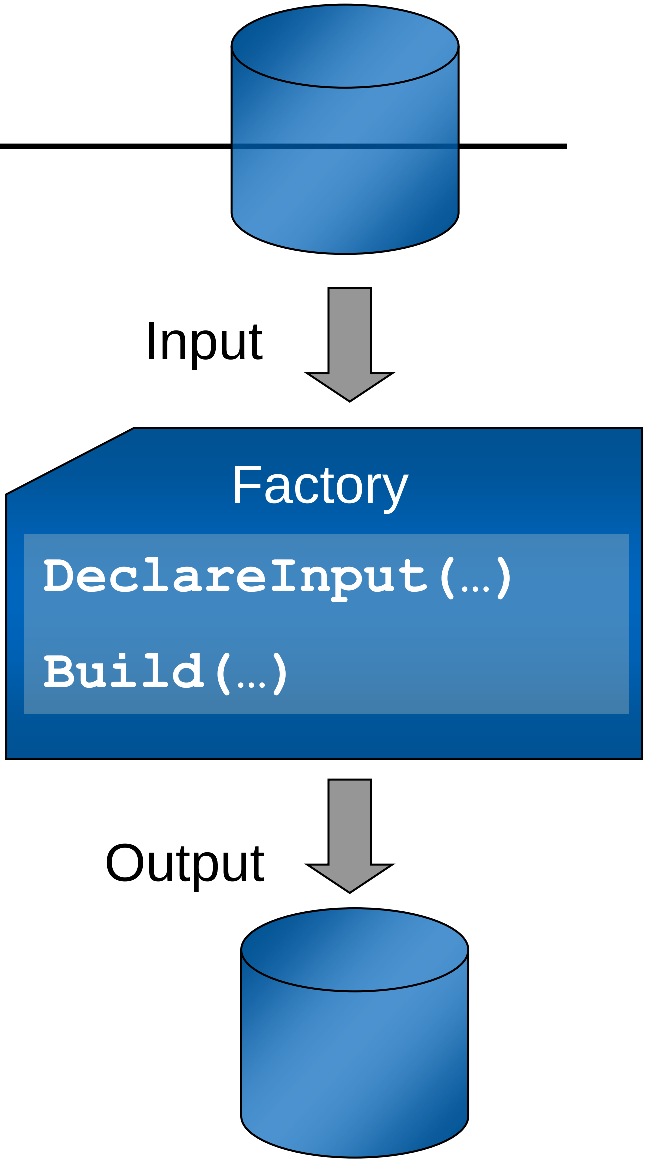
# Factories

- Factory processes input data (from Level) and generates some output data (stored in Level)

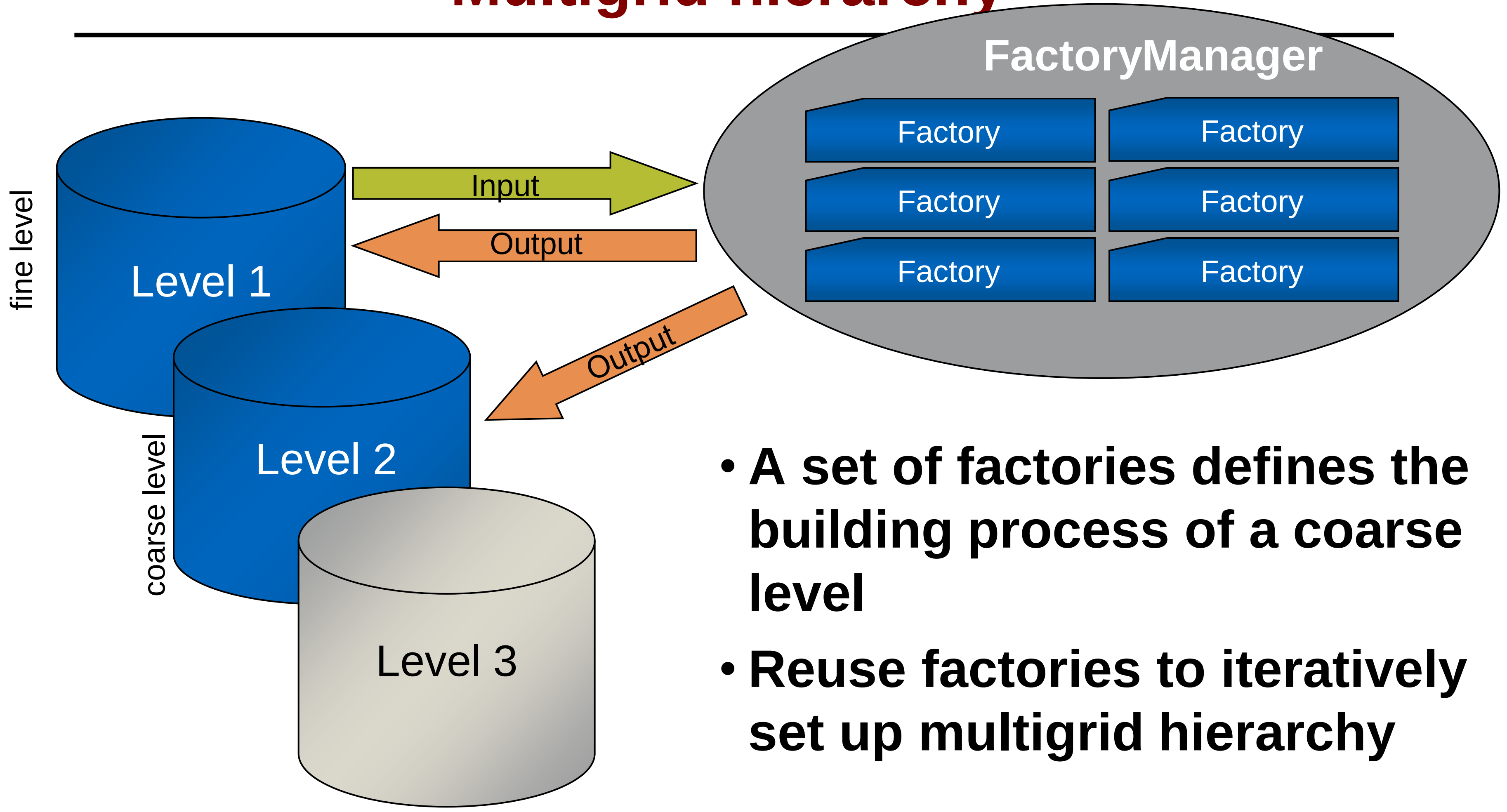
- Two types of factories
  - Single level (smoothers, aggregation, ...)
  - Two level (prolongators)

    Output is stored on next coarser level

Factory can generate more multiple output variables (e.g. „Ptent“ and „Nullspace“)

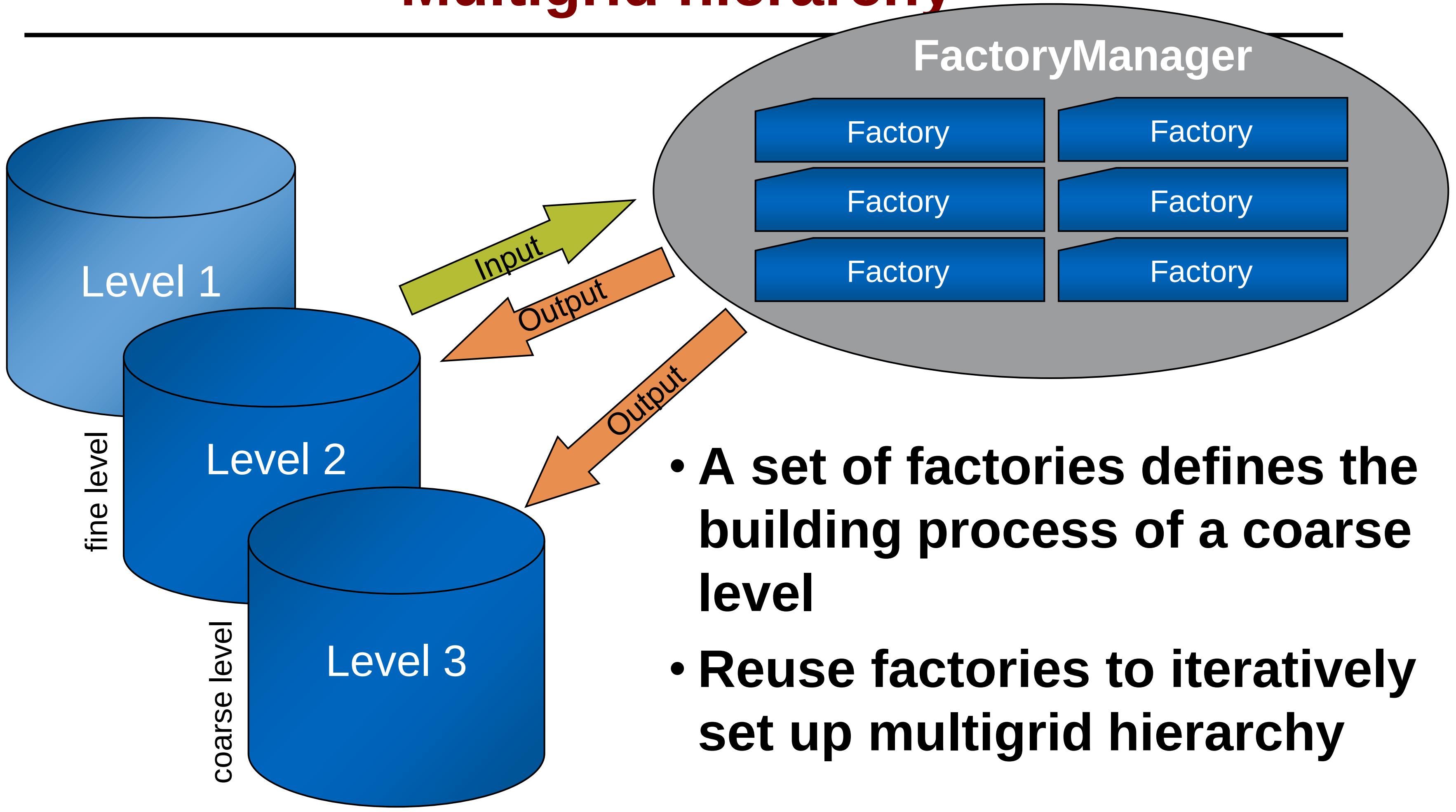


## Multigrid hierarchy



- A set of factories defines the building process of a coarse level
- Reuse factories to iteratively set up multigrid hierarchy

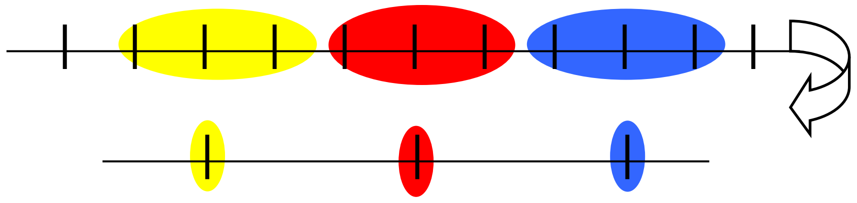
## Multigrid hierarchy



- **A set of factories defines the building process of a coarse level**
- **Reuse factories to iteratively set up multigrid hierarchy**

# Smoothed Aggregation Setup

- Group fine unknowns into *aggregates* to form coarse unknowns

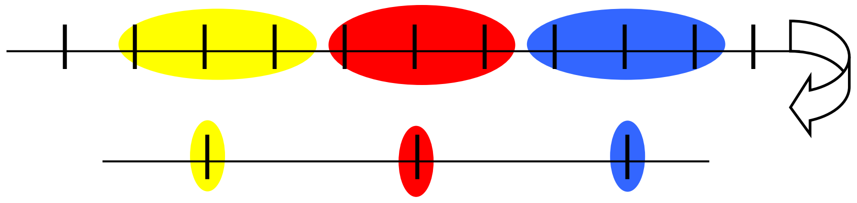


- Partition given nullspace $B_{(h)}$ across aggregates to have local support

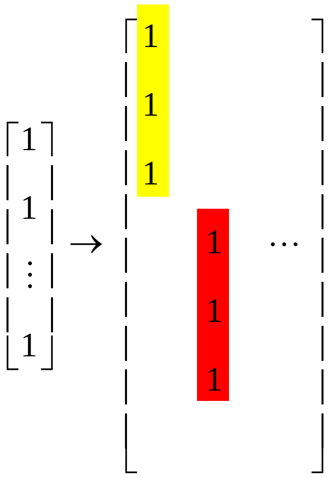
$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & & \\ & 1 & \cdots \\ & 1 & \\ & 1 & \\ & & \end{bmatrix}$$

# Smoothed Aggregation Setup

- Group fine unknowns into *aggregates* to form coarse unknowns



- Partition given nullspace $B_{(h)}$ across aggregates to have local support

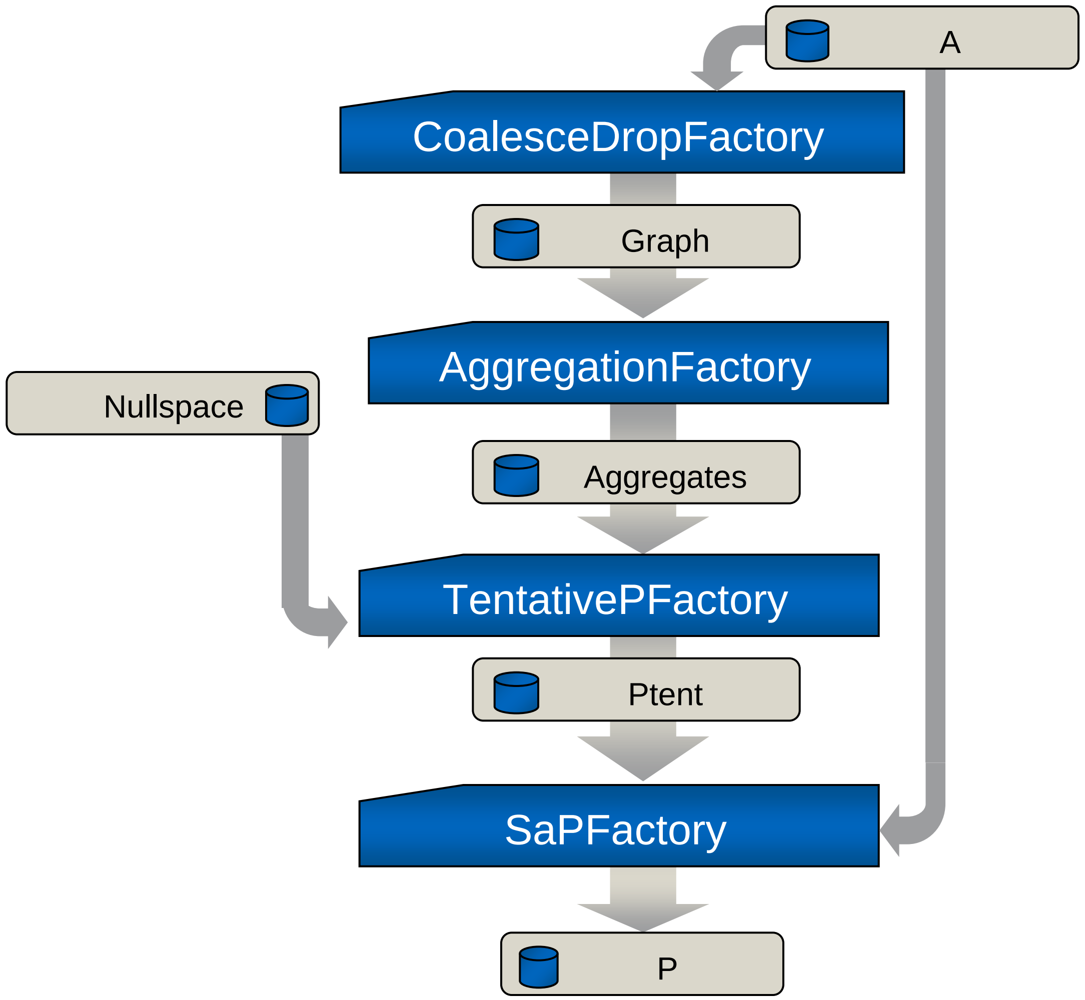


$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & & \\ & 1 & \cdots \\ & 1 & \\ & 1 & \\ & & \end{bmatrix}$$

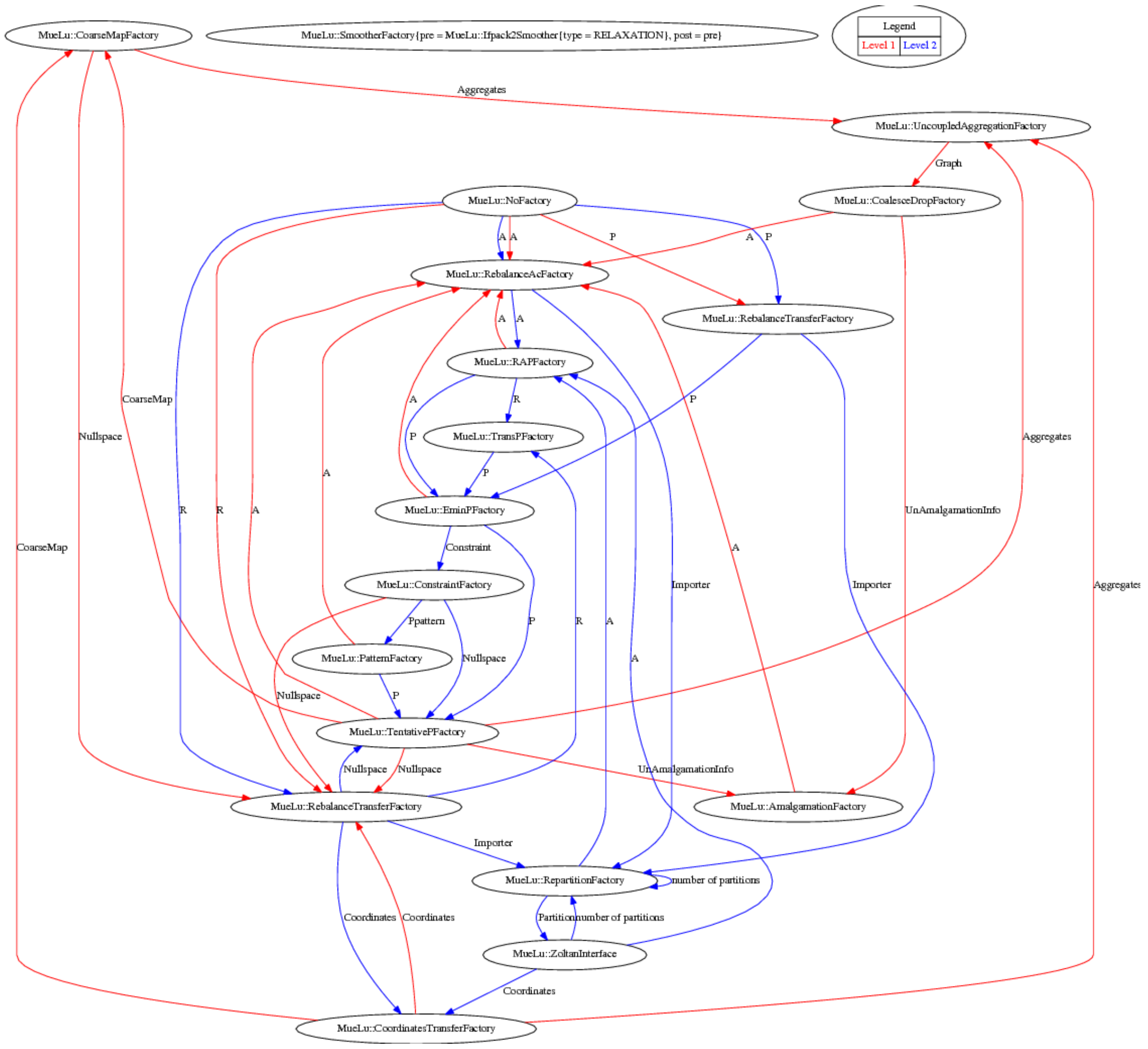
- Calculate $QR=B_{(h)}$ to get initial prolongator $P^{tent}$ (=$Q$) and coarse nullspace ($R$).

- Form final prolongator $P^{sm} = (I - \omega D^{-1}A)P^{tent}$

# Linking factories



# Linking factories



## User interfaces

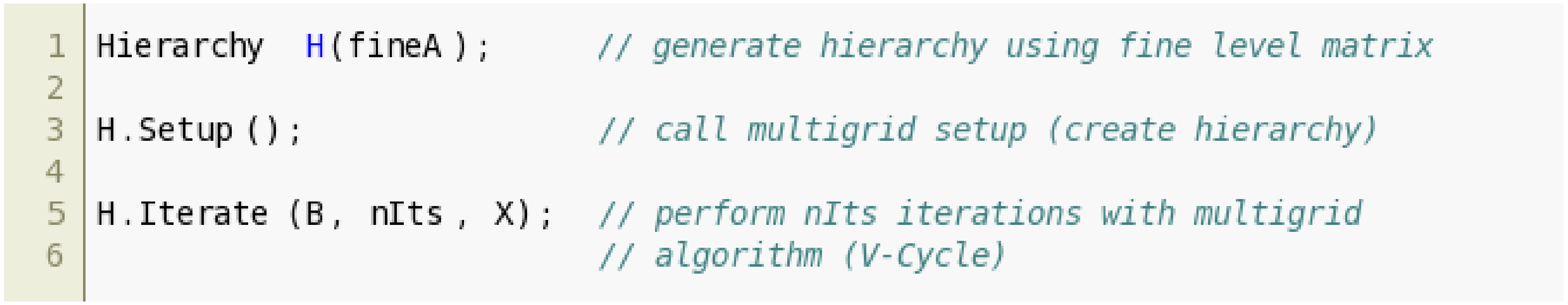
## MueLu – User Interfaces

- MueLu can be customized as follows:
  - XML input files
  - Parameter lists (key-value pairs)
  - Directly through C++ interfaces

- New/casual users
  - Minimal interface
  - Sensible defaults provided automatically

- Advanced users
  - Can customize or replace any component of multigrid algorithm.

# C++: smoothed aggregation



```
Hierarchy  H(fineA);      // generate hierarchy using fine level matrix

H.Setup();                // call multigrid setup (create hierarchy)

H.Iterate(B, nIts, X);    // perform nIts iterations with multigrid
                          // algorithm (V-Cycle)
```

- Generates smoothed aggregation AMG
- Uses reasonable defaults.
- Every component can be easily changed

## C++: unsmoothed aggregation

```
Hierarchy  H(fineA);                // generate hierarchy using fine level matrix

RCP<TentativePFactory > PFact = rcp(new TentativePFactory ());
FactoryManager  M;                  // construct factory manager
M.SetFactory ("P", PFact);          // define tentative prolongator factory
                                    // as default factory for generating P

H.Setup(M);                         // call multigrid setup (create hierarchy)

H.Iterate (B, nIts , X);            // perform nIts iterations with multigrid
                                    // algorithm (V-Cycle)
```

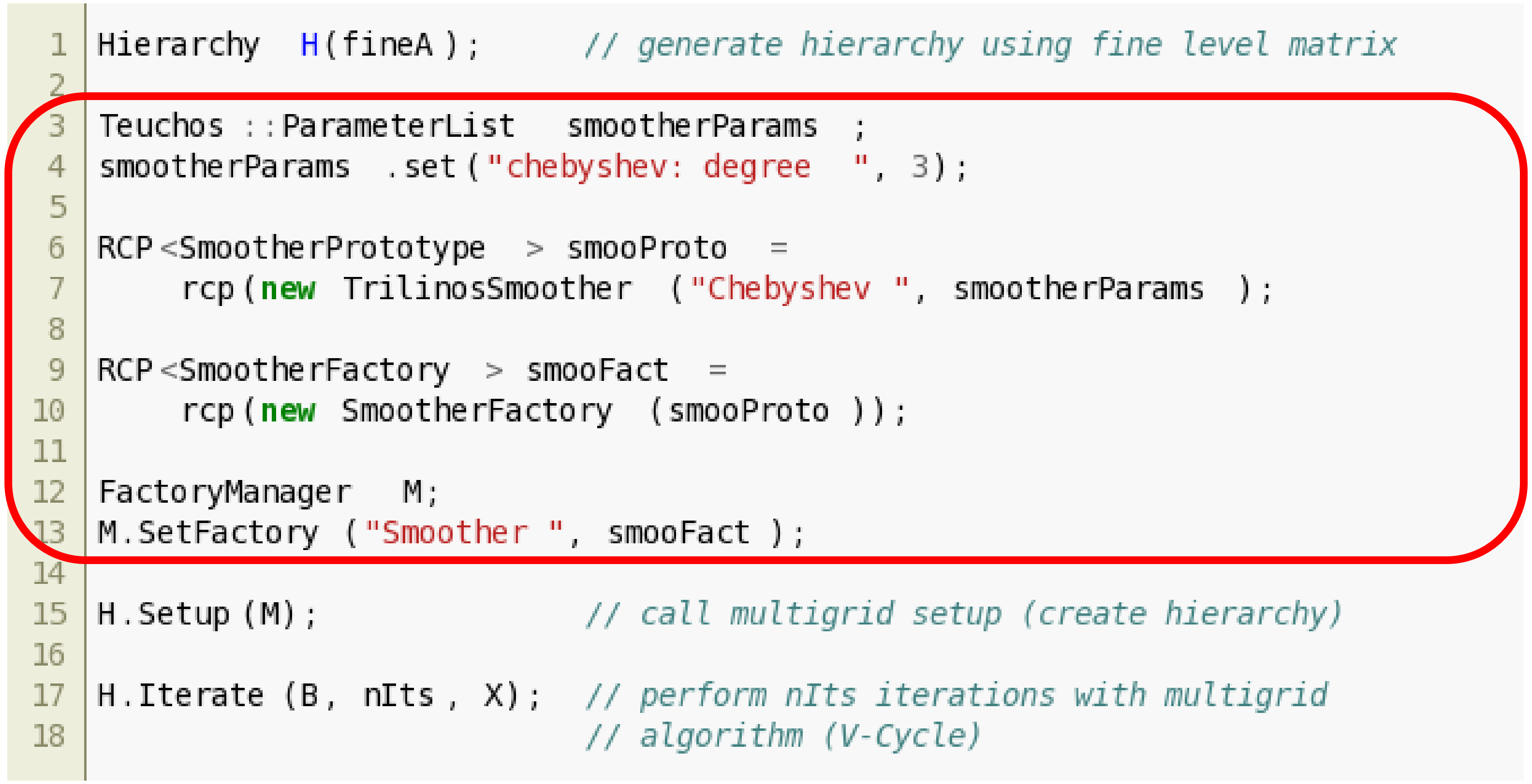
- **Generates unsmoothed prolongator**

# C++: unsmoothed aggregation

```
Hierarchy  H(fineA);                // generate hierarchy using fine level matrix

RCP<TentativePFactory> PFact = rcp(new TentativePFactory());
FactoryManager  M;                  // construct factory manager
M.SetFactory("P", PFact);           // define tentative prolongator factory
                                    // as default factory for generating P

H.Setup(M);                         // call multigrid setup (create hierarchy)

H.Iterate(B, nIts, X);              // perform nIts iterations with multigrid
                                    // algorithm (V-Cycle)
```

- **Generates unsmoothed prolongator**

# C++: polynomial smoother



```cpp
Hierarchy H(fineA);        // generate hierarchy using fine level matrix

Teuchos::ParameterList smootherParams;
smootherParams.set("chebyshev: degree", 3);

RCP<SmootherPrototype> smooProto =
    rcp(new TrilinosSmoother("Chebyshev", smootherParams));

RCP<SmootherFactory> smooFact =
    rcp(new SmootherFactory(smooProto));

FactoryManager M;
M.SetFactory("Smoother", smooFact);

H.Setup(M);                // call multigrid setup (create hierarchy)

H.Iterate(B, nIts, X);     // perform nIts iterations with multigrid
                           // algorithm (V-Cycle)
```

- **Uses degree 3 polynomial smoother**

## XML: creating hierarchy

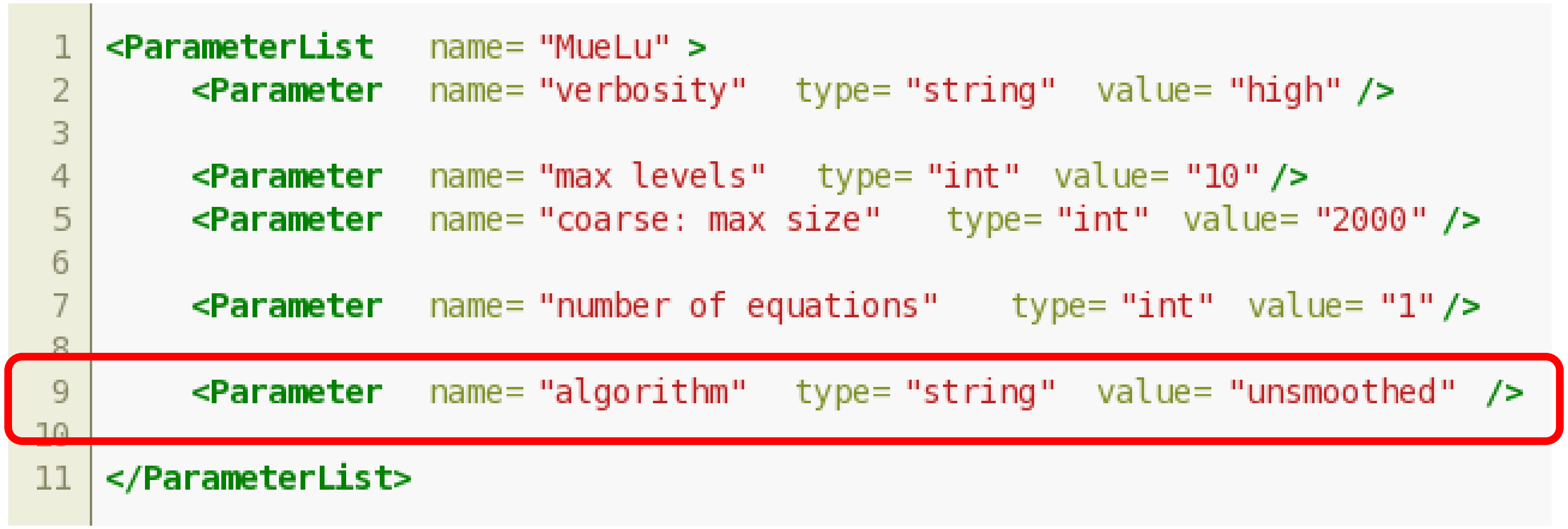
```
ParameterListInterpreter    mueluFactory (xmlFile );
RCP<Hierarchy > H = mueluFactory .CreateHierarchy  ();
H->GetLevel (0)->Set("A", fineA );

mueluFactory .SetupHierarchy  (*H);

H->Iterate (B, nIts , X);
```

## XML: smoothed aggregation

```xml
<ParameterList   name= "MueLu" >
     <Parameter  name= "verbosity"  type= "string"  value= "high" />

     <Parameter  name= "max levels"  type= "int"  value= "10" />
     <Parameter  name= "coarse: max size"   type= "int"  value= "2000" />

     <Parameter  name= "number of equations"   type= "int"  value= "1" />

     <Parameter  name= "algorithm"  type= "string"  value= "sa" />

</ParameterList>
```

- **Generates smoothed aggregation AMG**
- **Uses reasonable defaults**

## XML: unsmoothed aggregation



```
<ParameterList   name= "MueLu" >
     <Parameter  name= "verbosity"   type= "string"   value= "high" />

     <Parameter  name= "max levels"   type= "int"   value= "10" />
     <Parameter  name= "coarse: max size"    type= "int"   value= "2000" />

     <Parameter  name= "number of equations"    type= "int"   value= "1" />

     <Parameter  name= "algorithm"   type= "string"   value= "unsmoothed"   />

</ParameterList>
```

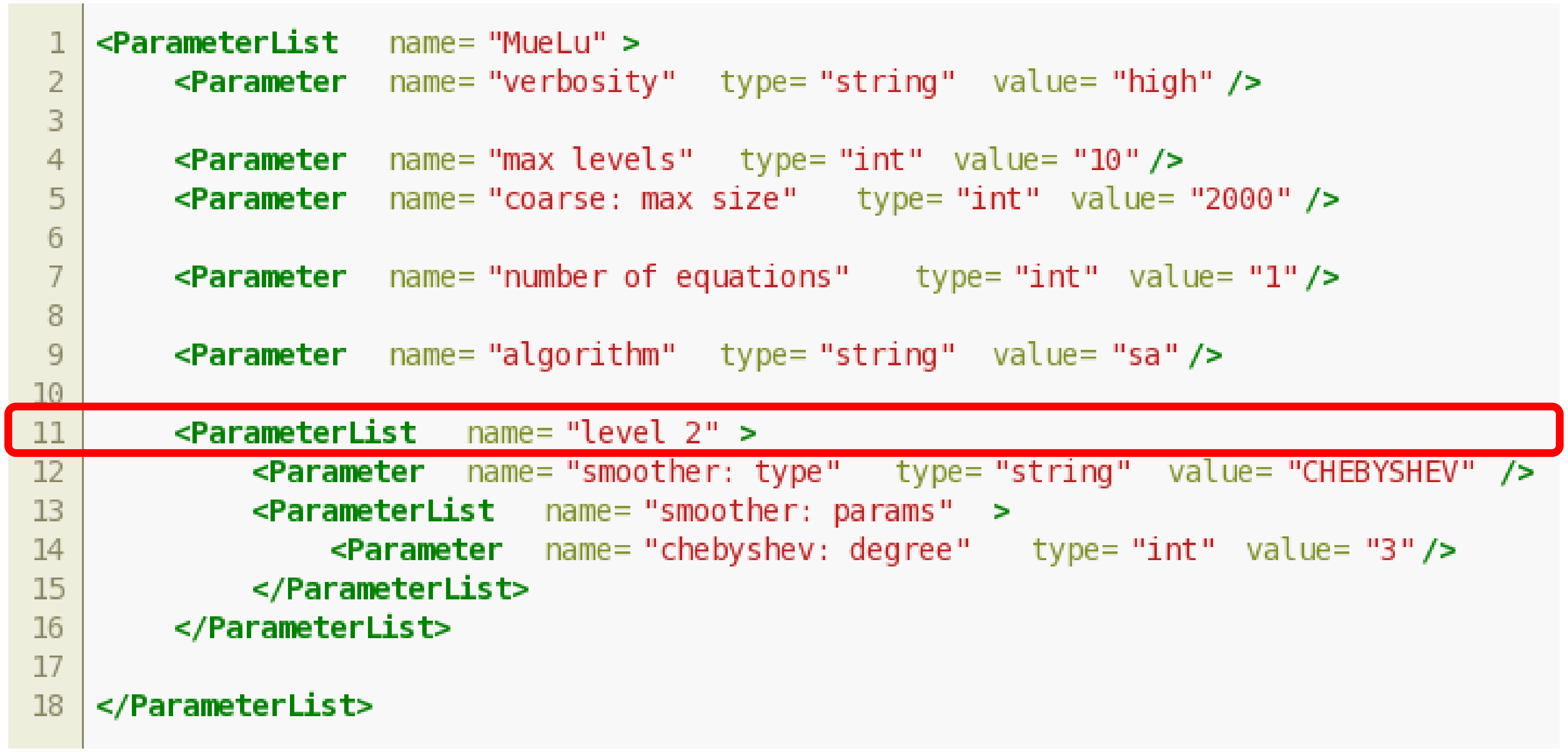
- **Generates unsmoothed prolongator**

# XML: polynomial smoother

```xml
<ParameterList  name="MueLu" >
    <Parameter  name="verbosity"  type="string"  value="high" />

    <Parameter  name="max levels"  type="int"  value="10" />
    <Parameter  name="coarse: max size"  type="int"  value="2000" />

    <Parameter  name="number of equations"  type="int"  value="1" />

    <Parameter  name="algorithm"  type="string"  value="sa" />

    <Parameter        name="smoother: type"  type="string"  value="CHEBYSHEV"  />
    <ParameterList    name="smoother: params"  >
        <Parameter      name="chebyshev: degree"  type="int"  value="3" />
    </ParameterList>

</ParameterList>
```

- **Uses degree 3 polynomial smoother**

# XML: polynomial smoother only for level 2



```
<ParameterList   name= "MueLu" >
    <Parameter   name= "verbosity"   type= "string"   value= "high" />

    <Parameter   name= "max levels"   type= "int"   value= "10" />
    <Parameter   name= "coarse: max size"    type= "int"   value= "2000" />

    <Parameter   name= "number of equations"    type= "int"   value= "1" />

    <Parameter   name= "algorithm"   type= "string"   value= "sa" />

    <ParameterList   name= "level 2" >
        <Parameter   name= "smoother: type"   type= "string"   value= "CHEBYSHEV"  />
        <ParameterList   name= "smoother: params"   >
            <Parameter   name= "chebyshev: degree"    type= "int"   value= "3" />
        </ParameterList>
    </ParameterList>

</ParameterList>
```

- **Uses degree 3 polynomial smoother for level 2**
- **Uses default smoother (Gauss-Seidel) for all other levels**

## Summary

- Current status
  - Part of publicly available Trilinos anonymous clone
  - We still support ML.

- Ongoing/Future work
  - Preparing for public release
    - Improving documentation
    - Improving application interfaces
  - Improving performance
  - Integrating existing algorithms
  - Developing new algorithms

Sandia National Laboratories